社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS2003-40 (2004-1)

マルチメディアシステムにおけるTimeliness QoS一貫性検証と時間制御コード導出

森　一夫†　岡野　浩三†　谷口　健一†

† 大阪大学大学院情報科学研究科　〒560-8531 豊中市待兼山町 1-3
E-mail: †{kazuo-mr,okano,taniguchi}@ist.osaka-u.ac.jp

あらまし　分散環境におけるマルチメディアシステムのコンポーネントに対する提供 Timeliness QoS とシステム全体における要求 Timeliness QoS を線形制約式で表現し，要求 Timeliness QoS が提供 Timeliness QoS の下で満たされること（一貫性）を検査する手法を提案する．提案手法では，この QoS 検査問題を線形計画法の非可解性問題に帰着させる．また検査済の QoS 一貫性を保証したコードを時間オートマトン表現を介して導出する手法についても述べる．
キーワード　Timeliness QoS，線形不等式，検証，時間オートマトン，導出

Consistency Checking of Timeliness QoS of Multi-Media Systems and Derivation of Codes for Timing Control

Kazuo MORI†, Kozo OKANO†, and Kenichi TANIGUCHI†

† Graduate School of Information Science and Technology, Osaka University
Machikaneyama 1-3, Toyonaka-shi, 560-8531 Japan
E-mail: †{kazuo-mr,okano,taniguchi}@ist.osaka-u.ac.jp

Abstract　This paper provides a method to check consistency of Timeliness QoS of a distributed multi-media system consisting of several components. In the method, provided/required timeliness QoS are represented in linear constraints. The method translates a timeliness QoS decision problem into an in-feasibility problem of linear programming. The paper also describes how to generate codes to control timing of each component of system.
Key words　timeliness QoS, linear inequality, verificationd, timed automata, derivation

1. はじめに

本稿では線形制約式で表現された Timeliness QoS の一貫性を検証する方法を提案し，時間オートマトンからその時間制約を満たした実行コードを導出する方法も提案する．

QoS（Quality of Service）の管理はコンポーネントベースのシステム設計において重要である．コンポーネントベースの設計法の一つに，各コンポーネントに対して QoS を仕様記述を与える方法がある [1]．この試みは特に UML [10] を用いて設計する場合に注目されている [11]．とりわけ，OMG（Object Management Group）ではリアルタイムシステムをサポートする研究を行われており，その中では QoS 仕様記述に UML プロファイル [11] を用いている．一般に，この方法の有用性の一つは，システム全体として期待される性能を満たすように，分散環境における各コンポーネントごとに適切な QoS 特性を記述できることである．一方，分散オブジェクトシステムの QoS を記述するために QML（QoS Modeling Language）という記述言語が開発されている．QML は QoS のカテゴリをユーザ定義の型として宣言し，さまざまな QoS を記述することができる．また分散環境を考慮して，コンポーネント間のインターフェイスで要求される QoS と提供すべき QoS をそれぞれ記述することができる．いずれのアプローチにおいても，コンポーネントの QoS 記述に対するシステム全体の要求 QoS との整合性，一貫性の検証に関する研究は十分になされていない．

QoS のうち時間に関するものを Timeliness QoS という．コンポーネントや，システム全体に関する Timeliness QoS の正しさの検証については，テストオートマトン [4] のアイデアに基づいて，文献 [12], [13] において，分散システムを構成するコンポーネントに対して記述された Timeliness QoS の影響を調べる形式的なアプローチを示している．各コンポーネントの振る舞いは時間オートマトンネット [3] でモデル化される．他に，CTL などの論理式を用いて検証を行なう方法も一般的である．これらのアプローチに共通する問題点は大規模なシステムに対しては，メモリ使用量の面から検証が困難になる点である．

そこで本稿では，システムが複数のコンポーネントからなると仮定し，それらの各コンポーネントに提供 Timeliness QoS を課す．この提供 Timeliness QoS がシステム全体が満たすべき要求 Timeliness QoS を満たしているかどうかという問いに対して，これらの QoS を時間変数を用いた線形制約式として記述し，線形制約の可解性判定問題に帰着し判定するアプローチをとる．このアプローチの利点は，コンポーネントを提供 Timeliness QoS で抽象化することにより，各コンポーネントを時間オートマトンなどで陽に記述する方法に比べてスケーラビリティが増すことにある．一般に，Timeliness QoS を信号 x の i 番目の発生時刻 x_i を用いて表すと，i については全称子をもった論理式となる．全称子を残したまま線形計画法を適用することはできない．一方，全称子をなくして元の意味通りの式を生成しようとすると無限個の式が必要となる．しかし実際には必要十分な有限個の式があれば，判定することは可能である．したがって提案手法では，Timeliness QoS の時間特性などを利用し，全称子を持たない有限個の線形制約式に変換し，判定する．

本稿の後半では，時間制御コード導出について述べる．仕様からのコード導出においては種々のモデルについてコード削減[14] など多くの手法が提案されている．システム全体の要求 Timeliness QoS が満たされることがわかれば，設計者は提供 Timeliness QoS を満たすコンポーネントの設計のフェーズに移る．提供 Timeliness QoS を満たすようにプログラムコードやハードウェア記述を直接記述する方法も考えられるが以下のような欠点をもつ．

コンポーネントの QoS 実現部と機能部の両方を記述する作業は非常に煩雑である．また実時間処理を扱うアプリケーションでは，複数の動作を並行して実行できる機構や，ある実行を指定した時間内に完了できる機構などが必要となる．このような制御の記述も煩雑であるため，コーディングの際にその提供 Timeliness QoS が保証されなくなることがある．

そこで，本研究では中間仕様として時間オートマトンでこれらを記述し，そこから時間制御用コードを自動生成するアプローチをとる．各コンポーネントに対し時間制約及び内部動作部を記述した時間オートマトンから，時間管理部と動作部をそれぞれ自動生成する．本手法では，時間管理はコンポーネントごとに行うことで自律性を実現している．しかし，この方法では各コンポーネントがクロックの異なる環境に配置された場合，システム全体として時間制御が正しく行われないことがあり得る．そこでシステムに新たにクロック調整モジュールを生成し，定期的にグローバルクロックを調節することで解決する．

以後，2. では Timeliness QoS について述べる．3. ではシステムに要求されている QoS の一貫性検証について説明し，4. では適用例について述べる．5. では時間オートマトンから時間制御用コードへの自動導出について説明し，6. でまとめる．

2. Timeliness QoS

QoS のうち，時間に関するものを Timeliness QoS という．本研究ではこのうちスループット，ジッタ，遅延の 3 つを扱う．

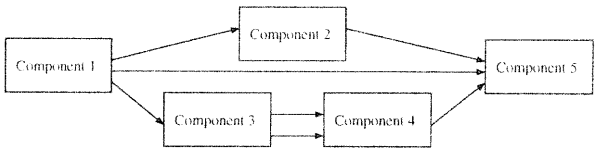

図 1　コンポーネントの接続関係
Fig. 1　Relations of components

以降では Timeliness QoS を単に QoS と呼ぶ．

図 1 はコンポーネントと信号の接続関係を表している．一般に，一つのコンポーネントにはコンポーネントによって定まる n 個の入力および m 個の出力信号 $\mathrm{x}^k(1 \leqq k \leqq n+m)$ が接続しているとする．なお，信号はこのように上付き添字で区別するか，あるいは，誤解がなければ，x，y，z などの記号を用いることとする．各信号 x^k に対して，$x_i^k(i>0)$ という変数を導入する．これは信号 x^k の i 番目の発生時刻（例えばフレーム出力信号における i 番目のフレームの出力時刻）を表す．曖昧さを招かない限り，添字 k を略し，x_i と表す．なお，x^k で信号 x^k の発生時刻系列を意味することとする．

2.1　線形制約式による Timeliness QoS の表現

本研究では検証時に，QoS については時間変数を介した表現（QoS 式）を用いて表現する．各 QoS 式は全称子が添字 i を束縛している．以下で各 QoS に対する QoS 式を示す．

2.1.1　スループット

ある期間 T に信号 x が少なくとも K 回発生しなければならないという制約は次のように QoS 式で表現される．

$$\forall i \in \mathbb{N}:\ x_{i+K-1} - x_i \leqq T$$

同様に，ある期間 T に信号 x が高々 K 回発生しなければならないという制約は次のように表現される．

$$\forall i \in \mathbb{N}:\ x_{i+K-1} - x_i \geqq T$$

なお，この形で表現されるスループットは Non-Anchored スループットと呼ばれる [6]．

2.1.2　ジッタ

T 間隔で発生する信号 x のジッタ制約の QoS 式は次のように表現される．

$$\forall i \in \mathbb{N}:\ T - m \leqq x_{i+1} - x_i \leqq T + M\ \ (m, M: \text{定数})$$

なお，この形で表現されるジッタは Non-Anchored ジッタと呼ばれる．これと対になる概念である Anchored ジッタは

$$\forall i \in \mathbb{N}:\ iT - m \leqq x_{i+1} \leqq (i+1)T + M\ \ (m, M: \text{定数})$$

で表現されるが，本稿ではこのタイプは扱わない．

2.1.3　遅　　延

2 つの信号 x と y の遅延関係が高々 T であるという制約を表す QoS 式は次のように表現される．

$$\forall i \in \mathbb{N}:\ 0 < x_i - y_{f(i)} \leqq T\ \ (f: i \text{についての関係式})$$

2.1.4　信号に対する仮定

信号系列 x に対して，以下の 2 つの性質を仮定する [3]．

（1）　単調増加性 $\forall i \in \mathbb{N}:\ x_i < x_{i+1}$

（2）　Non-Zenon 性 $\forall K > 0:\ \exists i:\ x_i > K$

3. システム要求 QoS の一貫性検証

3.1 検証方法のあらまし

システム全体として満たすべき QoS の集合を要求 QoS，システムを構成する各コンポーネントが提供する QoS の集合を提供 QoS と呼ぶ．以下の方法で要求 QoS の提供 QoS に対する一貫性を検証する方法について述べる．システム要求 QoS の QoS 式集合と各コンポーネントの提供 QoS の QoS 式集合，および，コンポーネントの接続関係から有限個の線形制約式からなる一貫性検証式を生成する．このとき，全称子があるため，QoS 式から変数置き換えなどの単純な方法で線形制約式を生成することはできず，いくつかの制約式を加えなければならない．以降では，一般性を失うことなく，システム要求 QoS 式は一つ，コンポーネント提供 QoS 式は複数与えられることとする．

3.2 検証可能な QoS とシステムのクラス

本手法で検証可能な QoS とシステムのクラスを示す．

QoS 式は，式中には変数が 2 つだけ存在し，その 2 変数の添字は $(i, i+K)$ または (i, Ki) の関係に限定されている（K はその式にのみ依存する定数）．また (i, Ki) の関係が用いられるのは遅延に関する QoS 式のみである．

一貫性判定の制限として，以下の 2 つを課す．

- 要求 QoS がジッタ制約であるとき，どこかのコンポーネントでジッタ制約が書かれていなければならない．
- 要求 QoS がスループット制約であるとき，どこかのコンポーネントでスループット/ジッタ制約が書かれていなければならない．かつ，それらのスループット制約における添字の差は，要求制約のそれの約数でなくてはならない．

最初の制限はなくしても検証することができるが，その結果は無意味なものとなる．これはジッタ制約からスループット制約を推論することはできるが，逆はできないためである．例えば，$\forall i \in \mathbb{N}:\ m \leqq x_{i+1} - x_i \leqq M$ から $\forall i \in \mathbb{N}:\ x_{i+10} - x_i \leqq 10M$ は容易に推論できるが（発信間隔が M 以内ならば，10 回目の発信時刻は最初の発信時刻からは $10M$ 以内に起こる），逆は推論できない（10 回目の発信時刻が最初の発信時刻からは $10M$ 以内に起こるからといって，発信間隔が M 以内であるとは言えない）．同様に最後の制限なしでも検証することはできるが，やはり結果は無意味なものとなる．

システムは複数のコンポーネントによって構成される．これはコンポーネントを頂点，信号の接続関係を辺とみた場合，有向グラフとみなすことができる．本研究で扱うシステムは図 1 のように閉路のない有向グラフとしてみなせるものに制限する．

3.3 検証手法

与えられた QoS 式と接続関係から線形制約の検証式を生成する方法を述べる．QoS 式には全称子が付いており，任意の発生時刻についての制約を表している．本検証手法では，QoS 式に付加された全称子を除去した式について検証を行う．一般には一つの QoS 式から，複数の相当する線形制約式を導出し，検証を行う．本検証手法では次のような検証式を生成する．

$$\left(\bigwedge \text{提供 } QoS \text{ 式から生成された式}\right)$$
$$\wedge \neg\left(\bigwedge \text{要求 } QoS \text{ 式から生成された式}\right)$$

この式は直観的に，要求 QoS 式を満たさないような場合があるかどうかを意味する論理式である．このため一貫性を満たすかどうかの真偽と検証式の真偽は反転する．つまり，検証式が偽を返すときに限り一貫性は保証される．

前節で述べた QoS 式制限の下で検証式を生成する方法とその方法の妥当性を簡単に示す．

システムを S を (V, E) と定義する．ここで V はコンポーネント集合（システムの入力/出力地点を表す特殊な要素 In, Out を含む），$E = V \times V$ はインターフェイス（信号の接続関係）集合である．また QoS 式の集合を C とし，各制約は

$$v_1 \leqq \alpha - \beta \leqq v_2$$

と定義される．ここで v_1, v_2 は定数で，α，β は (e, ix)(e は信号名，ix は添字) で表される信号発生時刻変数とする．特にスループット/ジッタ制約については，β 側の添字を信号の基準添字，β と α の添字の差を範囲 r とする．遅延式の場合は両方の添字を対応する信号の基準添字と呼ぶ．要求 QoS 式の集合は $Re \subset C$ で表し，提供 QoS 式の集合は $Pr \subset C$ で表す．このとき $Re \cup Pr = C$，$Re \cap Pr = \emptyset$ である．また写像 R を $V \cup E \to C$ と定義する．これはあるコンポーネント/インターフェイスに関する制約集合を表す写像である．

このとき検証式を生成する関数 generateFormulae() を次に示す．関数中に現われる $c.\alpha.e$ は QoS 式 c で使われる信号発生時刻変数 α の信号名 e を表す．また関数 Trace() はシステム出力から制約 c までに現われる遅延制約式を推移的にたどったときに現われる添字関係 (i, Ki) の K の総積を表す．

```
function generateFormulae(S)
for each c ∈ Pr
  T ← (c に現われる信号名，基準添字，範囲)
   c について全称子を外した線形制約式を生成 for
each {c | c ∈ R(e) ∧ e = (v, Out) ∈ E}
  if c の添字関係が (i, i + K) then
    // (c.α.e, p, q) が T に存在
    T ← T ∪ (c.β.e, p + K, q)
  else // c の添字関係が (i, Ki)
    // (c.α.e, p, q) が T に存在
    T ← T ∪ (c.β.e, Kp, Kq)
  for each システムの最後のコンポーネント v
    generateCompFormulae(v)
```

この関数では，まず全ての要求 QoS 式に現われる信号名，基準添字，範囲の 3 項組をテーブル T に登録する．システム出力につながるインターフェイスに関する遅延制約があった場合，遅延式に使われる添字によってずれが生まれる．例えば，あるコンポーネントの出力 x と入力 y の間に遅延制約

$$0 \leqq x_i - y_{i+2} \leqq 10$$

があるとき，このコンポーネントの出力以降で現われる添字 i に対し，入力以前の添字 i には何の関係もなく，$i+2$ が関係する．式生成時にはこのずれを考慮しなければならない．そこで，

この関数ではシステム出力/入力で使われる添字と各コンポーネントのQoS式で使われる信号変数の添字とのずれをテーブルに保持している．最後にシステムの最後のコンポーネント，すなわちコンポーネントの出力が他のコンポーネントではなく，直接システムの出力に接続されているものについてgenerateCompFormulae() を呼び出している．generateCompFormulae() を次に示す．

function generateCompFormulae(v)
if v がシステムの最後のコンポーネント or 全てのコンポーネント $v'(v' \mid (v,v') \in E)$ が式を生成済 **then**
  for each 遅延制約と出力側のスループット/ジッタ制約 c
    c に相当する範囲 n までの制約式を生成 ... (*)
  if 遅延の制約 c が存在 **then**
    if c の添字関係が $(i, i+K)$ **then**
      $T \leftarrow T \cup (c.\beta.e,$ 基準添字$, n)$
    else // c の添字関係が (i, Ki)
      // $(c.\alpha.e, p, q)$ が T に存在
      $T \leftarrow T \cup (c.\beta.e, Kp, Kq)$
  for each 入力側のスループット/ジッタ制約 $c \in R(v)$
    c に相当する範囲 n までの制約式を生成 ... (*)
  for each コンポーネント $v'(v' \mid (v', v) \in E)$
    generateCompFormulae(v')

関数 generateCompFormulae() は引数に与えられたコンポーネントからシステムの最初のコンポーネントまでの全てのコンポーネントについて制約式を生成する再帰関数である．この関数では，まず処理するコンポーネントの出力インターフェイス先のコンポーネントが全て処理済かどうかを検査し，処理済ならば，自コンポーネントの式生成を行う．このコンポーネントが複数の出力インターフェイスを持つ場合，各インターフェイス先のコンポーネントから複数回呼び出されるために，最後のインターフェイス先から呼び出されたときのみ生成処理を行うことを意味する．関数内部では，まずコンポーネントの出力信号のスループット/ジッタ制約式と遅延制約式について生成し，次に遅延式があればgenerateFormulae() と同じ方法でテーブルに追加する．それからコンポーネントの入力信号の制約式について生成する．自コンポーネントについて全ての制約式を生成した後で，入力インターフェイス先の全てのコンポーネントについてgenerateCompFormulae() を再帰呼出しする．

式生成 (*) について簡単に述べる．(*) が処理される時点で生成する制約式で使われる信号名，基準添字，範囲の組がテーブルに既に登録されている．これはスループット/ジッタの場合は1組であるし，遅延の場合は2組である．以下ではスループットの場合についてのみ説明するが，他の場合でも考え方は同じである．制約式が $v_1 \leqq x_{i+k} - x_i \leqq v_2$ であり，テーブルに (x,j,n) が登録されているとする．このとき，$v_1 \leqq x_{j+k} - x_j \leqq v_2$

$v_1 \leqq x_{j+2k} - x_{j+k} \leqq v_2$

$v_1 \leqq x_{j+3k} - x_{j+2k} \leqq v_2$

...

$v_1 \leqq x_{j+mk} - x_{j+(m-1)k} \leqq v_2$ $(mk = n)$

を生成する．

3.4 検証方法の正しさ

$\forall i \in \mathbb{N}:\ m \leqq x_{i+K} - x_i \leqq M$

が成り立っている場合に

$\forall i \in \mathbb{N}:\ m \leqq x_{i+K+1} - x_{i+1} \leqq M$

も成り立つ（$\forall$ の意味定義より）．一般に任意の0以上の定数 c について

$\forall i \in \mathbb{N}:\ m \leqq x_{i+K+c} - x_{i+c} \leqq M$

が成り立つ．このことと，時間に関する加算性を考慮にいれると任意の0以上の定数 n について以下が成立する．

$$\forall i \in \mathbb{N}:\ nm \leqq x_{i+nK} - x_i \leqq nM \quad (1)$$

次に

$\forall i \in \mathbb{N}:\ m \leqq y_{f(i)} - x_i \leqq M$

の式に付いて考える．$f(i)$ の式を $i+K$，ただし K は0以上の任意の定数と制限をおく．このときやはり

$\forall i \in \mathbb{N}:\ m \leqq y_{i+K+1} - x_{i+1} \leqq M$

さらには任意の0以上の定数 c について

$$\forall i \in \mathbb{N}:\ m \leqq y_{i+K+c} - x_{i+c} \leqq M \quad (2)$$

が成り立つ．しかし，この場合は，時間に関する加算性を考慮にいれても，以下の式は導出できない．「任意の0以上の定数 n について $\forall i \in \mathbb{N}:\ nm \leqq y_{i+nK} - x_i \leqq nM$」

一般にこの式を導出するためには x_i と y_i の関係式と x_i と x_{i+K} の関係式の2つが必要である．

[定義 1] QoS式の線形表現式

QoS式の全称子を無視し，添字付き変数 x_i を通常の変数 x に置き換えて得られる式（ただし，異なる添字を持つ変数は別変数に置き換える）をもとのQoS式の線形表現式と呼ぶ．

[命題 1] 要求QoS式において一般性を失うことなく使用されている変数リテラルが x_i, x_{i+K} であり，かつ提供QoSの集合に変数リテラル x_i, x_{i+L} に関する式があると仮定する．ただし L は K の約数とする．generateFormulae() で生成される式が解を持たないときかつそのときに限り，提供QoSは要求QoSを満たす．

[略証 1] 変数 x_i，x_{i+K} に対応する線形表現式上の変数を，一般性を失うことなく x，x' と表す．提供QoS式から変数 x，x' に関する，式 (1) を導くのに必要十分な線形表現式がgenerateFormulae() の最後のfor each文で生成される．必要十分な式が生成される保証はそれぞれの方向の証明で与える．一方，要求QoSの x，x' に関する，式 (1) と同じ形の線形表現式がgenerateFormulae() の最初のfor each文で生成される．前者の線形表現式の許容解空間を Pr，後者のそれを Re とする．

⇒: 最終的に生成される式に解がないと仮定する．このとき Pr が Re の部分集合である．すなわち要求QoSの線形表現が表す変数 x，x' の許容解空間が提供QoSから生成される線形表現の対応する変数の許容解空間を含む．このとき線形表現式の定義より，提供QoSが要求QoSを満たしている．なお，generateFormulae() で生成される線形表現式はもとのQoS式のインスタンスとみなすことが各式の生成方法から確認することができる．これにより最終式の健全性が保証される．

⇐: 提供 QoS が要求 QoS を満たすと仮定する．仮定と以下の理由により，変数 x，x' の許容解に対して Pr は Re の部分集合となる．

（1）変数 x，x' の関係が一つのコンポーネントについて閉じている場合は添字の範囲と基準点の選び方が generateCompFormulae() で保証されていることが確認できる．これと (1) 式，(2) 式を導出した方法をあわせて十分性が保証できる．

（2）変数 x，x' の関係が複数のコンポーネントにわたる場合は，generateFormulae() では x' から x に至る全てのコンポーネントのパスについて，関係添字 $ni+m$ の必要な組み合わせに関して変数の連立不等式を作成していることが確認できる．従って，最終的に生成される式が偽となる． □

なお，生成される式の有限性も容易に確かめることができる．上記の議論は (i, Ki) の QoS 式および (x_i, y_j) の関係式についても同様にできる．ただし後者については generateFormulae() において，x と y の関係式は基準点と基準点からの範囲に関する 2 式のみを生成することによって対応する．

3.5 生成される線形制約表現式のサイズ

generateFormulae() では，各 QoS 式に対し対応する複数の線形制約式を生成している．各 QoS 式に対しどれくらいの線形制約式が生成されるかについて述べる．

まず遅延式について述べる．遅延式の一般形は

$\forall i \in \mathbb{N}:\ 0 < x_i - y_{f(i)} \leqq T$ (f : i についての関係式)

である．generateFormulae() がシステム出力側から逆向きにたどりながら行われているために，遅延 QoS 式の線形制約式を生成する際には，テーブルには信号 x について (信号名 x，基本添字 j，範囲 n) が存在する．このとき生成される式は

$$0 < x_j - y_{f(j)} \leqq T,\quad 0 < x_{j+n} - y_{f(j+n)} \leqq T$$

の 2 つだけでよい．

次にスループット/ジッタ式について述べる．ただしジッタについてはスループット式の一般形

$\forall i \in \mathbb{N}:\ T' \leqq x_{i+K-1} - x_i \leqq T$

において $K = 2$ の場合であるのでスループット式についてのみ述べる．スループット QoS 式について線形制約式を生成する際，やはりテーブルには信号 x についての (信号名 x，基本添字 j，範囲 n) が存在する．このとき generateFormulae() の (*) の部分よりもとの QoS 式の範囲を n' とすると n/n' 個が生成される．この値は generateFormulae() より，遅延 QoS 式についての 3 項組が生成された generateCompFormulae が呼び出されるまでの系列内で現れた遅延制約式のうち，添字関係が (i, Ki) であるものの K の総積に等しい．

以上より生成される線形制約表現式の個数は，QoS 式の数 n，全ての遅延制約式のうち添字関係が (i, Ki) であるものの K の総積 $k = \prod K$ とするとき $O(kn)$ で抑えられる．

4. 適用例題

検証を行う例題として，図 2 のようなビデオプレイヤーシステムを考える．このシステムは Reader，Buffer，Subtitle，Composer という 4 つのコンポーネントから構成される．Reader はビデオストリームを生成し，パケット信号を出し，Buffer は Reader からパケットを受け取り Composer に送るバッファである．Subtitle はビデオストリームに付加する字幕を生成する．最後に Composer が Buffer と Subtitle からそれぞれパケットと字幕を受け取りフレームを生成する．この例では簡単のために，1 パケットが 1 フレームから生成されるとする．

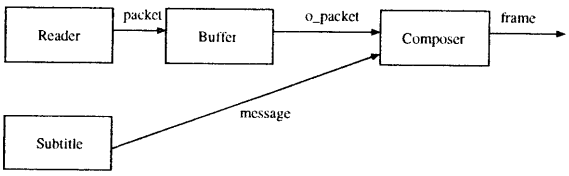

図 2 例題：ビデオプレーヤーシステム
Fig. 2 Examle: Video Player System

表 1 各コンポーネントの提供 QoS
Table 1 Provided QoS for each component

コンポーネント名	提供 QoS
Reader	$\forall i \in \mathbb{N}:\ 100 \leqq RdrO_{i+1} - RdrO_i \leqq 150$
Buffer	$\forall i \in \mathbb{N}:\ 0 \leqq BufO_i - RdrO_i \leqq 400$
Subtitle	$\forall i \in \mathbb{N}:\ 500 \leqq SubO_{i+5} - SubO_i$
Composer	$\forall i \in \mathbb{N}:\ 50 \leqq ComO_i - BufO_i$ $\forall i \in \mathbb{N}:\ 50 \leqq ComO_i - SubO_i$

システム要求 QoS とコンポーネント提供 QoS は表 1 とする．各 QoS を簡単に説明する．Reader では出力信号のスループットを，Subtitle では出力信号のジッタをそれぞれ提供している．これはそれぞれ内部の動画や字幕データをフレーム用に分割したものを出力するために最低限の処理時間を要するが，処理時間に制限を課していることを意味する．Buffer は遅延の QoS を提供している．現時点ではバッファ容量などは考慮せず，単に Buffer にパケットが入ってから出るまでの時間制約を記述している．Composer は 2 種類の信号を受け取り，そのデータを合成したものをフレームとして出力する．そのため出力時刻は 2 つの入力に依存するので，2 つの遅延制約を提供している．本来ならばコンポーネント間の通信部分（インターフェイス）にも遅延時間がかかるがこの例では簡単のために遅延が 0 とし，2 つのコンポーネント間の出力信号と対応する入力信号の名前を同じにしている．このときシステム要求 QoS：

$\forall i \in \mathbb{N}:\ 1200 \leqq SysO_{i+10} - SysO_i \leqq 1500$

の検証式を生成した．この QoS はプレイヤーシステムのスループット制約を記述している．

先に述べたように，全称子を除いた式だけでは検証することができず，この例では新たに，

- $100 \leqq RdrO_{i+K} - RdrO_{i+K-1} \leqq 150\ (K = 2, 3, ..., 10)$
- $0 \leqq BufO_{i+10} - RdrO_{i+10} \leqq 400$
- $500 \leqq SubO_{i+10} - SubO_{i+5}$
- $50 \leqq ComO_{i+10} - BufO_{i+10}$
- $50 \leqq ComO_{i+10} - SubO_{i+10}$

を生成することで検証が可能となる．

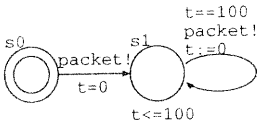

図 3 Reader コンポーネントの時間オートマトン
Fig. 3 Timed Automata for Reader component

5. 時間制御用コード導出方針

5.1 導出のあらまし

時間オートマトンから時間制御コードを導出するが，このとき各コンポーネントの時間オートマトンは先に検証された提供 QoS を満たすように設計・検査されているものとする．この検査には文献 [4], [12], [13] などの方法が使える．導出されるコードには，時間管理用モジュールが含まれており，そのモジュールが自コンポーネントの提供 QoS を満たすために内部クロックによって時間制御する．

5.2 時間オートマトンの概要

時間オートマトン [3] は実時間アプリケーションの動作仕様を形式的に記述できるモデルの一つである．時間オートマトンは，有限オートマトンに任意個のクロックとクロック制約を付加したものである．

例題で用いたシステムの Reader コンポーネントの時間オートマトンを図 3 に示す．この時間オートマトンは 2 つの状態をもち，s0 が初期状態であり，更にクロック変数 t を持っている．クロック変数は単位時間ごとにその値が 1 ずつ増加する．状態 s0 ではイベント packet を発生し，状態 s1 に遷移する．s1 に遷移すると状態に付加された制約を満たす限りその s1 にとどまることができる．s1 では遷移に付加された条件式（この例では t==100）を満たしているときに遷移することができる．

5.3 導出コードの実行方式

目的コードは各コンポーネントごとに導出され，時間制御を行うモジュール（管理モジュール）と，時間オートマトンに対応するモジュール（TA モジュール）をもつ．さらに全体の時間管理を定期的に行うクロック調整モジュールも導出する．

実行時に TA モジュールは現時点でアクティブな状態を保持している．また管理モジュールへのインターフェイスとして現状態の取得，変更を提供する．TA モジュールは管理モジュールにより制御され，新しい状態に遷移すると，その状態に対応する実行コードを実行する．ただし，本導出手法は時間オートマトンから時間制御部のみを導出するので，実行コードには何も記述されておらず導出後，設計者が新たに記述する．

管理モジュールは時間オートマトンで使われているクロック変数を監視し，遷移のタイミングをスケジュールする．内部データとして，時間オートマトンの遷移関係，クロック変数による条件式と式と遷移との対応を保持している．管理モジュールは TA モジュールから現状態を取得し，その状態から起こり得る遷移に対応した遷移条件式を選び出す．それから保持しているクロックを監視し，条件式を満たしていれば式に対応する遷移を候補として保持する．何らかの遷移を行う場合には TA モジュールにアクセスし，現状態を変更する．

6. あとがき

複数のコンポーネントからなる分散マルチメディアシステムに対し，それらの各コンポーネントに提供 QoS を課す．この提供 QoS がシステム全体が満たすべき要求 QoS を満たしているかどうかを問う問題に対して，これらの QoS を線形制約式として記述し，線形制約の可解性判定問題に帰着することにより，QoS の一貫性を検査する方法論について述べた．また，時間オートマトンで記述された各コンポーネントの動作仕様から時間制御コードを導出する方法について述べた．

これらの検証/導出系を製作し，例題に適用し評価することが今後の課題である．また，一貫性検査可能なクラスの限界を調べることも課題の一つである．

文　献

[1] R. Staehli, F. Eliassen, J. Aagedal and G. Blair "Quality of Service Semantics for Component-Based Systems," 2nd Int'l Workshop on Reflective and Adaptive Middleware Systems, pp. 153–157, 2003

[2] D. H. Akehurst, B. Bordbar, J. Derrick and A. G. Waters: "Design Support for Distributed Systems: DSE4DS," In J.Finney, M.Haahr, and A.Montressor, editors, Proceedings of the 7th Cabernet Radicals Workshop, October 2002.

[3] R. Alur and D.L. Dill: "A Theory for Timed Automata," In Theoretical Computer Science 125 pp.183–235, 1994.

[4] L. Ageto, P. Bouyer, A. Burgueño and K. G. Larsen: "The Power of Reachability Testing for Timed Automata," In Proceedings of 18th Conference of Fundamental of Software Technology and Theoretical Computer Science (FST and TCS '98), LNCS1530, pp.245–256, 1998.

[5] K. G. Larsen, P. Pettersson and W. Yi: "UPPAAL in a Nutshell," In Springer International Journal of Software Tools for Technology Transfer 1(1+2) 1997.

[6] H. Bowman, G. Faconti and M. Massink: "Specification and verification of media constraints using UPPAAL," In Proceedings of Design, Specification and Verification of Interactive Systems '98, Markopoulos and P. Johnos, editors, pp. 261–277 Springer, 1998.

[7] G. Blair and J.-B. Stefani: "Open Distributed Processing and Multimedia," Addison-Wesley, Boston, MA, 1997.

[8] B. Bordbar, J. Derrick and A. G. Waters: "A UML approach to the design of open distributed systems," In Chris George and Huaikou Miao, editors, Formal Methods and Software Engineering, LNCS2495, pp.561–572, 2002.

[9] J. R. Putman: "Architecting with RM-ODP," Prentice Hall. Upper Saddle River, NJ, 2001.

[10] "UML The Unified Modelling Language Version 1.4," Object Management Group(http://www.omg.org).

[11] UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, Request for Proposal, available at http://www.omg.org

[12] B. Bordbar and K. Okano, "Verification of Timeliness QoS Properties in Multimedia Systems," 5th International Conference on Formal Engineering Methods (ICFEM '03), LNCS 2885, pp. 523–540, 2003

[13] 加藤雄一郎，山口弘純，岡野浩三，谷口健一，"拡張時間オートマトン群による実時間システムの記述および検証，" 電子情報通信学会技術研究報告，Vol. 102, No. 616, pp. 13–18, 2003.

[14] 坂上弘祐，岡野浩三，谷口健一，"ペトリネットで記述された簡易ブラウザ型の組込み Java プログラム動作仕様に対する実行方式の提案，" 電子情報通信学会技術研究報告，Vol. 102, No. 246, pp. 1–6, 2002.